

## NVM Express Technical Errata

<b>Errata ID</b>	<b>011</b>
<b>Change Date</b>	<b>5/11/2011</b>
<b>Affected Spec Ver.</b>	<b>NVM Express 1.0a</b>
<b>Corrected Spec Ver.</b>	

### Submission info

Name	Company	Date
Peter Onufryk	IDT	4/26/2011
Kevin Marks	Dell	4/26/2011

This erratum makes editorial changes to section 5.

This erratum makes editorial changes and clarifications to section 7.5 on interrupts.

Throughout the document, “completion entry” is changed to “completion queue entry”, including the plural form.

**Modify section 5.7.1 as shown:**

A completion **queue** entry is posted to the Admin Completion Queue **when if** the controller has **completed complicated** the requested action (specified in the Activate Action **parameter field**). Firmware Activate command specific errors are defined in Figure 45.

**Figure 45: Firmware Activate – Command Specific Errors Values**

Value	Description
6h	<b>Invalid Firmware Slot:</b> The firmware slot indicated is invalid <b>or read only</b> . This error is indicated if the firmware slot exceeds the number supported.
7h	<b>Invalid Firmware Image:</b> The firmware image specified for activation is invalid and not loaded by the controller.

**Modify section 7.5 as shown below:**

The interrupt architecture allows for efficient reporting of interrupts such that the host may service interrupts through the least amount of overhead.

The **specification allows the** controller **may to** be configured to report interrupts in one of four modes. The four modes are: pin-based interrupt, single message MSI, multiple message MSI, and MSI-X. It is recommended that MSI-X be used whenever possible to enable higher performance, lower latency, and lower CPU utilization for processing interrupts.

Interrupt aggregation, also referred to as interrupt coalescing, mitigates host interrupt overhead by reducing the rate at which interrupt requests are generated by a controller. This reduced host overhead typically comes at the expense of increased latency. Rather than prescribe a specific interrupt aggregation algorithm, this specification defines the mechanisms a host may use to communicate desired interrupt aggregation parameters to a controller and leaves the specific interrupt aggregation algorithm used by a controller as vendor specific. Interrupts associated with the Admin Completion Queue should not be delayed.

The Aggregation Threshold field in the Interrupt Coalescing feature (refer to section 5.12.1.8) **indicates specifies** the host desired minimum interrupt aggregation threshold on a per vector basis. This value defines the number of Completion Queue entries that when aggregated on a per interrupt vector basis reduces host interrupt processing overhead below a host determined threshold. This value is provided to the controller as a recommendation by the host and a controller is free to generate an interrupt before or after this aggregation threshold is achieved. The specific manner in which this value is used by the interrupt aggregation algorithm implemented by a controller is implementation specific.

The Aggregation Time field in the Interrupt Coalescing feature (refer to section 5.12.1.8) **indicates specifies** the host desired maximum delay that a controller may apply to a Completion Queue entry before an interrupt is signaled to the host. This value is provided to the controller as a recommendation by the host and a controller is free to generate an interrupt before or after this aggregation time is achieved. A controller may apply this value on a per vector basis or across all vectors. The specific manner in which this value is used by the interrupt aggregation algorithm implemented by a controller is implementation specific.

**Modify section 7.5.1 as shown below:**

This is the mode of interrupt operation if any of the following conditions are met:

- Pin based interrupts are being used – MSI (MSICAP.MC.MSIE='0') and MSI-X are disabled
- Single MSI is being used – MSI is enabled (MSICAP.MC.MSIE='1'), MSICAP.MC.MME=0h, and MSI-X is disabled
- Multiple MSI is being used – Multiple-message MSI is enabled (MSICAP.MC.MSIE='1') and (MSICAP.MC.MME=1h) and MSI-X is disabled.

Within the controller there is an interrupt status register (IS) that is not visible to the host. In this mode, the IS register determines whether the PCI interrupt line shall be driven active or an MSI message shall be sent. Each bit in the IS register corresponds to an interrupt vector. The IS bit is set to '1' when the AND of the following conditions is true:

- There is one or more unacknowledged completion **queue** entries in a Completion Queue that utilizes this interrupt vector;
- The Completion Queue(s) with unacknowledged completion **queue** entries has interrupts enabled in the "Create I/O Completion Queue" command;
- The corresponding INTM bit exposed to the host is cleared to '0', indicating that the interrupt is not masked.

If MSIs are not enabled, ~~the resulting IS bits are logical "ORed" together – any bit IS[0]~~ being a one causes the PCI interrupt line to be active (electrical '0'). If MSIs are enabled, any change to the IS register that **causes an unmasked status bit to transition from zero to one or clearing of a mask bit whose corresponding status bit is set does not result in the register being all cleared** shall cause an MSI to be sent. Therefore, while in wire mode, a single wire remains active, while in MSI mode, several messages may be sent, as each edge triggered event on a port shall cause a new message.

In order to clear an interrupt for a particular interrupt vector, **host** software shall acknowledge all completion **queue** entries for Completion Queues associated with the interrupt vector.

Status of IS Register	Pin-based Action	MSI Action
All bits '0' <b>Note:</b> May be caused by corresponding bit(s) in the INTM register being set to '1', masking the corresponding interrupt.	Wire inactive	No action
One or more bits set to '1' <b>Note:</b> May be caused by corresponding bit(s) in the INTM register being cleared to '0', unmasking the corresponding interrupt.	Wire active	New message sent
One or more bits set to '1', new bit gets set to '1'	Wire active	New message sent
One or more bits set to '1', <del>software clears</del> some (but not all) bits in the IS register are cleared (i.e., <b>host software acknowledges some of the associated completion queue entries</b> )	Wire active	No action
One or more bits set to '1', <del>software clears</del> all bits in the IS register are cleared (i.e., <b>host software acknowledges all associated completion queue entries</b> )	Wire inactive	No action

**Modify the first paragraph of section 7.5.1.1 as shown below:**

It is recommended that host software utilize the **Interrupt Mask Set and Interrupt Mask Clear (INTMS/INTMC)** registers to efficiently handle interrupts when configured to use pin based or MSI messages. Specifically, within the interrupt service routine, host software should set the appropriate mask register bits to '1' to mask interrupts via the INTMS register. In the deferred procedure call, host software should process all completion **queue** entries and acknowledge the completion **queue** entries have been processed by writing the associated CQyHDBL doorbell registers. When all completion **queue** entries have been processed, host software should unmask interrupts by clearing the appropriate mask register bits to '0' via the INTMC register.

It is recommended that the MSI interrupt vector associated with the CQ(s) being processed be masked during processing of completion **queue** entries within the CQ(s) to avoid spurious and/or lost interrupts. For single message or multiple message MSI, the INTMS and INTMC registers should be used to appropriately mask interrupts during completion **queue** entry processing.

**Modify section 7.5.1.1.1 as shown below:**

An example of the host software flow for processing interrupts is described in this section. This example assumes multiple message MSI is used and that interrupt vector 3 is associated with I/O Completion Queue 3.

1. The controller posts a completion **queue** entry to I/O Completion Queue 3. The controller sets IS[3] to '1' in its internal IS register. The controller asserts an interrupt to the host.
2. The interrupt service routine (ISR) is triggered.
3. Host software scans all I/O Completion Queues **associated with the asserted MSI vector** to determine the location of new completion **queue** entries. In this case, a new completion **queue** entry has been posted to I/O Completion Queue 3.
4. Host software writes 08h to the INTMS register to mask interrupts for interrupt vector 3, which is associated with I/O Completion Queue 3.
5. The controller masks interrupt vector 3, based on the host write to the INTMS register.
6. Host software schedules a deferred procedure call (DPC) to process the completed command.
7. The deferred procedure call (DPC) is triggered.
8. Host software processes new completion **queue** entries for I/O Completion Queue 3, completing the associated commands to the OS. Host software updates CQyHDBL to acknowledge the processed completion **queue** entries and clear the interrupt associated with those completion **queue** entries. If all completion **queue** entries have been acknowledged by host software, the controller de-asserts interrupt vector 3.
9. Host software unmasks interrupt vector 3 by writing 08h to the INTMC register.

**Modify the last two paragraphs of section 7.5.1.2 as shown below:**

For multiple MSI, the controller advertises the number of MSI interrupt vectors it requests in the Multiple Message Capable (MMC) field in the Message Signaled Interrupt Message Control (MC) register. The MSICAP.MC.MMC field represents a power-of-2 wrapper on the number of requested vectors. For example, if three vectors are requested, then the MSICAP.MC.MMC field **must shall** be '010' (four vectors).

Multiple-message MSI allows completions to be aggregated on a per vector basis. **If sufficient MSI vectors are allocated, each** ~~Each~~ Completion Queue(s) may send its own interrupt message, as opposed to a single message for all completions.

**Modify section 7.5.2 as shown below:**

This is the mode of interrupt operation if the MSI-X is being used – (multiple-message) MSI is disabled (MSICAP.MC.MSIE='0') and (MSICAP.MC.MME=0h) and MSI-X is enabled. This is the preferred interrupt behavior to use.

MSI-X, similar to multiple-message MSI, allows completions to be aggregated on a per vector basis. However, the maximum number of vectors is 2K. MSI-X also allows each interrupt to send a unique message data corresponding to the vector.

MSI-X allows completions to be aggregated on a per vector basis. Each Completion Queue(s) may send its own interrupt message, as opposed to a single message for all completions.

When generating an MSI-X message, the following checks occur before generating the message:

- The function mask bit **in the MSI-X Message Control register** is not set to '1'
- The corresponding vector mask in the MSI-X table structure is not set to '1'

If either of the masks are set, the corresponding pending bit in the MSI-X PBA structure is set to '1' to indicate that an interrupt is pending for that vector. The MSI for that vector is later generated when both the mask bits are reset to '0'.

It is recommended that the interrupt vector associated with the CQ(s) being processed be masked during processing of completion **queue** entries within the CQ(s) to avoid spurious and/or lost interrupts. The interrupt mask table defined as part of MSI-X should be used to mask interrupts.

**Throughout the specification, make the following change:**

Change instances of “completion entry” to “completion queue entry”.

Change instances of “completion entries” to “completion queue entries”.

#### Disposition log

4/26/2011	Erratum captured.
5/11/2011	Clarified that pin based interrupts use IS[0].
6/20/2011	Erratum ratified.

*Technical input submitted to the NVMHCI Workgroup is subject to the terms of the NVMHCI Contributor's agreement.*